REMARKS

Claims 1-6, 22-27, 32-37, 53-58, 63-68 and 84-89 are pending.

Claims 4-5, 35-36, and 66-67 have been canceled in this response. Claims 1, 6, 32, 37, 63, and been amended.

Claims 63-68 and 84-89 are rejected under 35 U.S.C. §101 as directed to non-statutory subject matter. The specification has been amended to remove the reference to transmission-type media and, therefore, it is respectfully submitted that the rejection under 35 U.S.C. §101 has been overcome.

Claims 1-6, 22-27, 32-37, 53-58, 63-68, and 84-89 are rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,557,076 to Copeland ("Copeland"). Applicants respectfully traverse the rejections.

Applicant's invention relates to a method, system, and apparatus for fragment caching. Specifically, a fragment header is defined to be used within a network protocol, such as Hypertext Transfer Protocol (HTTP). The fragment header associates metadata with the fragment for various purposes related to processing and caching the fragment. In addition, cache ID rules accompany the fragment from its origin server. The cache ID rules describe a method for forming a unique cache ID for the fragment such that dynamic content can be cached away from its origin server. A dependency ID, which may differ from a cache ID, may be associated with the fragment such that a server may an invalidation operation that purges a fragment from a cache. More specifically, the present invention provides an indication that the fragment is non-cacheable to non-fragment-supporting cache management units and an indication that the fragment is cacheable to fragment-supporting cache management units.

In contrast, Copeland relates to relates to a method and apparatus for processing data stored in memory, such as cache memory. If data stored in the memory is invalidated, then a determination is made whether an indicator is associated with the data. If so, then the data is retrieved from its source without requiring an external request for the data. Those of skill in the art would recognize that both the present invention and Copeland implement a source indicator for the data. However, Applicants' invention provides an indication that the fragment is non-

cacheable to non-fragment-supporting cache management units and an indication that the fragment is cacheable to fragment-supporting cache management units, whereas the approach taught by Copeland does not. Furthermore, the approach of Copeland presumes that data is already stored in memory and uses a source indicator to refresh invalidated data.

Independent claims 1, 32, and 63 have been amended to recite that the message comprises an indication that the fragment is non-cacheable to non-fragment-supporting cache management units and an indication that the fragment is cacheable to fragment-supporting cache management units.

Regarding Claim 4, Examiner asserts that Copeland teaches the method of Claim 1, further comprising: determining that a message header in the message indicates that the fragment is cacheable (Col. 7, lines 66 – Col. 8, Line 5; Col. 9, lines 48-56). Applicants respectfully disagree with Examiner. The cited reference does not teach whether a fragment is, or is not, cacheable, but whether it should be cached. According, those of skill in the art would not consider the teachings to be equivalent.

Examiner asserts that Copeland teaches the aforementioned limitation citing Col. 9, lines 48-50. Applicants respectfully disagree with Examiner. The cited reference merely relates to providing an indicator as to whether or not a fragment should be cached. Copeland fails to teach the provision of an indicator that indicates the fragment is non-cacheable to non-fragment-supporting cache management units and cacheable to fragment-supporting cache management units.

Regarding Claim 2, Examiner asserts that Copeland teaches the method of Claim 1, further comprising: storing a fragment from a message in a cache maintained by a cache management unit within the computing device, wherein the cache management unit operates equivalently in support of fragment caching operations whether the computing device acts as a client, a server, or a hub throughout the network (Col. 9, lines 13-29). Applicants respectfully disagree with Examiner. While Applicants concede that the cited reference of Copeland relate to computing devices that are servers, clusters of servers, or nodes in a cluster, those of skill in the art would not consider these to be equivalent to a client.

Regarding Claim 3, Examiner asserts that Copeland teaches the method of Claim 1, further comprising: determining that a message header in the message indicates that a message body portion of the message is a fragment (Col. 7, line 66 – Col. 8, line 5; Col. 9, lines 42-56). Applicants respectfully disagree with Examiner. The cited reference contains no reference to a message header indicating that a message body portion of the message is a fragment. Instead, the cited reference relates to a source indicator (the "URL") of the fragment and providing and indicator to determine whether or not the fragment should be cached.

Regarding Claim 6, Examiner asserts that Copeland teaches the method of Claim 5, wherein the message comprises an HTTP cache-control header with a no-cache directive for non-fragment-supporting cache management units and with a directive for caching the fragment for fragment-supporting cache management units (Col. 15, lines 20-47). Applicants respectfully disagree with Examiner. The cited reference relates to the rendering of fragments in a cache and assumes that the fragment is already cached. It will be apparent to skilled practitioners of that the rendering of a fragment in a cache is not equivalent to providing a no-cache directive for non-fragment-supporting cache management units and a directive for caching the fragment for fragment-supporting cache management units.

Regarding Claim 22, Examiner asserts that Copeland teaches the method of Claim 1, further comprising: retrieving a set of dependency identifiers from the message, wherein a dependency identifier is generated by a server that originated the fragment (Col. 10, lines 7-25, data ID corresponds to the "dependency identifier"); and storing the set of dependency identifiers in association with a source identifier for the fragment (fragment ID) (Col. 10, lines 21-25).). Applicants respectfully disagree with Examiner. The cited reference relates to a Data ID, which may be used to represent the underlying data which cause the fragment to be invalidated (Col. 10, lines 10-11). In contrast, the Dependency ID of the present invention, which may differ from a cache ID or a URI for a fragment, may be associated with a fragment so that a server may initiate an invalidation operation that purges a fragment from a cache. Those of skill in the art would understand that representing which underlying data would cause the fragment to be invalidated is not equivalent to the initiation of an invalidation operation that purges a fragment from a cache.

Regarding Claim 23, Examiner asserts that Copeland teaches the method of Claim 22, further comprising: receiving an invalidation request message; retrieving a dependency identifier from the invalidation request message; determining a set of fragments that are associated with the dependency identifier; and purging the set of fragments from the cache in response to determining the set of fragments that are associated with the dependency identifier (Col. 10, lines 10-25; Col. 7, line 66 – Col. 8, line 5). Applicants respectfully disagree with Examiner. The cited reference relates to a Data ID, which may be used to represent the underlying data which cause the fragment to be invalidated. The cited reference fails to teach the retrieval of a dependency identifier from an invalidation request message, irrespective of the fact that the cited reference fails to teach the receipt of an invalidation request message itself. Accordingly, those of skill in the art would fail to see the equivalence between representing the underlying data which cause the fragment to be invalidated and retrieving a dependency identifier from an invalidation request message; determining a set of fragments associated with the dependency identifier; and purging the set of fragments that are associated with the dependency identifier from the cache.

Regarding Claim 24, Examiner asserts that Copeland teaches the method of Claim 1, further comprising: retrieving a set of fragment caching rules from the message, wherein a fragment caching rule determines a manner for generating a cache identifier for the fragment (Col. 9, line 13 – Col. 10, line 40); and generating a cache identifier fro the fragment in accordance with a fragment caching rule (Col. 11, lines 15-25). Applicants respectfully disagree with Examiner. The cited references refer to "allow(ing) the JSP to obtain a cache ID" (Col. 11, lines 18-19). The difference between obtaining a cache ID and generating cache identifier for the fragment would be obvious to skilled practitioners of the art and not considered equivalent.

Regarding Claim 25, Examiner asserts that Copeland teaches the method of Claim 24, further comprising uniquely identifying the cache identifier (Col. 15, lines 34-47). Applicants respectfully disagree with Examiner. The cited reference relates to the rendering of cache IDs, not the identification of cache identifiers taught by the present invention.

Regarding Claim 26, Examiner asserts that Copeland teaches the method of Claim 24, further comprising: performing the storing operation using the generated cache identifier for the

fragment (Col. 15, lines 19-47). Applicants respectfully disagree with Examiner. Since the cited reference fails to teach the generation of a cache identifier for the fragment, it would be obvious that there would be no cache identifier to store.

Regarding Claim 26, Examiner asserts that Copeland teaches the method of Claim 24, further comprising: obtaining at least a path portion of a Uniform Resource Identifier (URI) associated with the fragment in order to form a base cache identifier (Col. 11, line 65 – Col, 12, line 35; Col. 15, lines 19-47); and applying a fragment caching rule to the base cache identifier to form a cache identifier for the fragment, wherein a fragment caching rule comprises a set of query parameter names and/or cookie names that are used to obtain name-value pairs that are appended to the base cache identifier (Col. 11, lines 15-43; Col. 15, lines 19-47). Applicants respectfully disagree with Examiner. The cited references relate to the rendering of cache IDs, not the forming of a base cache identifier from at least a path portion of a URI. Accordingly, it would be apparent to those of skill in the art that it would be impossible to apply a fragment caching rule to the base cache identifier to form a cache identifier for the fragment since the cache identifier had never been formed.

The arguments submitted hereinabove also apply to the rejection of claims 32-34, 37, 53-58, 63-65, 68, and 84-89.

For the reasons set forth hereinabove, all pending claims are patentable over the art of record and the rejection of the pending claims under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,557,076 to Copeland should be removed.

CONCLUSION

In view of the amendments and remarks set forth herein, Applicants respectfully submit that all pending claims are in condition for allowance. Accordingly, Applicants request that a Notice of Allowance be issued. Nonetheless, should any issues remain that might be subject to resolution through a telephone interview, the Examiner is requested to telephone the undersigned at 512-338-9100.

CERTIFICATE OF TRANSMISSION

I hereby certify that on May 18, 2009, this correspondence is being transmitted via the U.S. Patent & Trademark Office's electronic filing system.

/Gary W. Hamilton/

Respectfully submitted,

/Gary W. Hamilton/

Gary W. Hamilton
Attorney for Applicant(s)
Reg. No. 31,834